Final Report

Learning with data adaptive features

By Yongdai Kim

Department of Statistics, Seoul National University, Korea

and Statistical Research Center for Complex Systems, Seoul National University, Korea

1. Summary

It is frequently observed that the dimension of inputs is much larger than the sample size. Examples are image construction, microarray data, data mining etc. In such cases, standard learning methods either are not applicable or perform badly. Also, identifying a small subset of important features, which discriminate outputs, becomes an important subject. Hence, good learning algorithms with high dimensional inputs should provides a classification rule which not only yields high accuracy but also has an ability of identifying few important features.

Apparently, there are two popularly used such algorithms. One is decision trees and the other is LASSO. The objective of this research is to develop new algorithms for decision tree and LASSO for improving computational power, prediction accuracy, and ability of detecting significant features.

2. Results

In this section, we summarize the research results. Details are appended.

For decision trees, we have developed a new pruning algorithm and showed that combining with the Bumping procedure, the proposed pruning algorithm improves prediction accuracy significantly. Moreover, its computing time is much less than that for the standard method of constructing decision trees since the proposed method does not need the cross-validation step which is usually computationally demanding.

For LASSO, we have developed a new computational algorithm based on the gradient descent method. The proposed algorithm does not require any sophisticated optimization

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **27 JUL 2006**	2. REPORT TYPE **Final Report (Technical)**	3. DATES COVERED **30-09-2003 to 30-01-2005**

4. TITLE AND SUBTITLE **Learning with Data Adaptive Features**	5a. CONTRACT NUMBER **F6256203P0658**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Yongdai Kim**	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Statistical Research Center for Complex Systems,Seoul National University,Seoul 151-477,KOREA,KE,151-477**	8. PERFORMING ORGANIZATION REPORT NUMBER **AOARD-034050**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **The US Resarch Labolatory, AOARD/AFOSR, Unit 45002, APO, AP, 96337-5002**	10. SPONSOR/MONITOR’S ACRONYM(S) **AOARD/AFOSR**
	11. SPONSOR/MONITOR’S REPORT NUMBER(S) **AOARD-034050**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
It is frequently observed that the dimension of inputs is much larger than the sample size. Examples are image construction, microarray data, data mining etc. In such cases, standard learning methods either are not applicable or perform badly. Also, identifying a small subset of important features, which discriminate outputs, becomes an important subject. Hence, good learning algorithms with high dimensional inputs should provides a classification rule which not only yields high accuracy but also has an ability of identifying few important features. Apparently, there are two popularly used such algorithms. One is decision trees and the other is LASSO. The objective of this research is to develop new algorithms for decision tree and LASSO for improving computational power, prediction accuracy, and ability of detecting significant features.

15. SUBJECT TERMS
Machine Learning, Ensemble Algorithms, Statistical Learning Theory

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**36**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

algorithms and hence it is much faster and stabler. Also, we derived the convergence rate of the proposed algorithm, and surprisingly it turns out that the convergence rate does not depend on the dimension of inputs.

3. Contributions to machine learning and future works

LASSO has gained popularity recently due to its superiority in accuracy as well as the sparsity. However, its computation requires special optimization techniques and so it is hardly applied to large dimensional data, in particular when the loss function is not the square loss. The proposed algorithm in this research resolves this computational issues completely, and hence LASSO techniques can be used freely to large dimensional data.

For future works, we can develop faster and simpler algorithm for LASSO and other types of sparse models such as fused LASSO and blockwise sparse model. The convergence rate of the proposed algorithm is m^{-1} where m is the number of iteration. We can improve it up to the exponential convergence.

The new pruning method for decision trees opens a new direction of finding good decision trees. In particular, when the proposed method can be used with ensemble algorithms such as boosting and random forest, we believe that more efficient and informative decision trees can be found. We leave this approach as a future work.

Publications

Kim, J. and Kim, Y. (2004). Maximum a posteriori pruning for decision trees and its application to Bumping. Accepted by *Computational Statistics and Data Analysis.*

Kim, Y. and Kim, J. (2004). Gradient LASSO for feature selection. In the Proceedings of the International Conference on Machine Learning, 2004.

Kim, J., Kim, Y. and Kim, Y. (2004). Gradient descent algorithm for generalized LASSO. under revision for *Journal of Machine Learning Research.*

Maximum a posteriori pruning on decision trees and its application to bootstrap BUMPing

Jinseog Kim [a,*] Yongdai Kim [b] Jongwoo Jeon [b]

[a] *Statistical Research Center for Complex Systems, Seoul National University, Seoul 151-742, Korea*

[b] *Department of Statistics, Seoul National University, Seoul 151-742, Korea*

Abstract

The cost-complexity pruning generates nested subtrees and selects the best one. However, its computational cost is large since it uses hold-out sample or cross-validation. On the other hand, the pruning algorithms based on posterior calculations such as BIC (MDL) and MEP are faster, but they sometimes produce too big or small trees to yield poor generalization errors. In this paper, we propose an alternative pruning procedure which combines the ideas of the cost-complexity pruning and posterior calculation. The proposed algorithm uses only training samples, so that its computational cost is almost same as the other posterior-based algorithms, and at the same time yield stable results as the cost-complexity pruning. Moreover it can be applied for comparing non-nested trees, which is necessary for the BUMPing procedure. Empirical results show that the proposed algorithm performs similarly as the cost-complexity pruning in the standard situation and works better for BUMPing.

Key words: cost-complexity pruning, decision trees, cross-validation, BIC, posterior probability, BUMPing

* Tel : 82-2-880-6987
Email addresses: `jskim@stats.snu.ac.kr` (Jinseog Kim), `ydkim@stats.snu.ac.kr` (Yongdai Kim), `jwjeon@plaza.snu.ac.kr` (Jongwoo Jeon).

1 Introduction

Decision trees are very popular models in applied statistics and machine learning for classification and regression (Breiman et al., 1984; Quinlan, 1993) since they are simple and easy to be interpreted. The typical procedure of the decision tree induction consists of two steps, so called growing and pruning. The growing step constructs a fully grown tree, which is usually done by finding the best split recursively until no more splits are possible. Then, the pruning step selects the best subtrees of the fully grown tree.

Various pruning algorithms have been developed such as the cost-complexity pruning(CCP; Breiman et. al, 1984), reduced-error pruning(REP; Quinlan, 1987), Bayesian Information Criterion (BIC) or equivalently minimum description length (MDL) pruning (Rissanen, 1978; Quinlan and Rivest, 1989; Mehta et al., 1995), minimum error pruning (MEP, Niblett and Bratko, 1986) and so on. These pruning algorithms can be classified into two types. The first type uses hold-out samples, to which CCP and REP belong to, and the second type including BIC/MDL and MEP uses only training samples. For the pruning methods of the first type, a sufficiently large amount of hold-out samples should be required for selecting the best tree. If hold-out samples are not available, one can resort to heuristic methods such as the V-fold cross validation(CV), but it demands more computations additionally and the final result is unstable. Moreover, the CV is not easily applicable for comparing non-nested trees. One such example is BUMPing (Tibshirani and Knight, 1999), which first constructs many decision trees using bootstrap samples and then searches the best model within them. Therefore, for BUMPing, a method for comparing non-nested decision trees is necessary. In contrast, the pruning methods of the second type do not pose these problems. However, the finally selected tree tends to be unnecessarily large or small, which results in lower prediction accuracy and arise difficulty in interpreting the model. See section 5 for empirical evidences.

The aim of this article is to develop a new pruning algorithm to resolve the aforementioned problems. The main idea of the proposed pruning algorithm is to combine the ideas of the CCP and Bayesian methods. The proposed algorithm first generates the sequence of nested trees in the same manner as CCP, and compares them via calculating their posterior probabilities to choose the best one. By doing so, we can reduce computing time and avoid the instability due to the CV. Also, we can easily compare non-nested trees by their posterior probabilities, and hence can improve the performance of BUMPing.

In order to evaluate the posterior probabilities of given trees, we could use BIC since it is known as an approximation of the posterior probability. However, for tree models, we observed that BIC sometimes but not often yields unnecessary large or small trees (see Table 3 in section 5). Mehta et al. (1995) reported the similar results. This is partly because BIC internally put equal prior probabilities on trees of different sizes and the degree of approximation is poor for large sized trees since the number of observations in each terminal node is not sufficiently large. The proposed algorithm overcomes these deficiencies in BIC by putting a size-dependent prior and calculating the exact posterior probabilities.

The paper is organized as follows. Section 2 briefly describes the three well known pruning algorithms - CCP, BIC and MEP. Section 3 explains the proposed pruning algorithm including the way of specifying prior probabilities and calculating posterior probabilities for given decision trees. Section 4 presents the BUMPing technique based on the proposed pruning algorithm. In section 5, we give the results of empirical studies using real data sets. Finally, discussions follow in section 6.

2 Review of Pruning Methods

In this section, we briefly review the three pruning methods based on CCP, BIC(MDL) and MEP, respectively. The objective of this paper is to combine the ideas of these pruning methods to develop a new pruning method.

2.1 Cost-complexity pruning(CCP)

Breiman et al. (1984) developed the minimal cost complexity pruning(CCP) which performs as follows :

(1) Construction of the set of candidate subtrees $T_0, T_1, \ldots, T_K$, according to the cost-complexity measures.
(2) Selection of the best tree T^* among $T_0, T_1, \ldots, T_K$ according to estimates of the generalization errors through V-fold CV or hold-out pruning samples.

Let $T_{\max}$ denote the fully grown tree for a given training data set. For any subtree T of $T_{\max}$ and $\lambda > 0$, Breiman et al. (1984) defined the cost complexity measure as follows :

$$R_\lambda(T) = R(T) + \lambda|\tilde{T}|,$$

where $R(T)$ is a misclassification error of T calculated on the training data, and $|\tilde{T}|$ is the number of terminal nodes in T. For a given λ, let T_λ be the optimal subtree of $T_{\max}$ which minimizes $R_\lambda(T)$. Breiman et al. (1984) showed that for the increasing values of $0 = \lambda_0 < \lambda_1 < \ldots$, the corresponding sequence of optimal subtrees $T_{\lambda_0}, T_{\lambda_1}, \ldots$ forms a nested structure. They proposed to estimate the optimal $\hat{\lambda}$ by minimizing the CV error or the hold-out sample error, and then the corresponding tree $T_{\hat{\lambda}}$ is selected as the final best tree.

2.2 Bayesian Information Criterion (BIC)

Let $M_1, M_2, \ldots$ be given models and let $\pi(M_1), \pi(M_2)$, ... be the prior probabilities. For a given model M, the likelihood and prior of parameters are denoted by $P(Y|\Theta, M)$ and $\pi(\Theta|M)$, respectively. Then the posterior probability of the model M can be computed by integrating out the parameter Θ. That is,

$$P(M|Y) \propto \pi(M) \int P(Y|\Theta, M)\pi(\Theta|M)d\Theta. \tag{1}$$

When the integral part of the above equation is very complex, it is not easy to obtain the closed form directly. However, we can approximate $P(M|Y)$ by

$$\log P(M|Y) \approx \log P(Y|\hat{\Theta}, M) - \frac{d}{2}\log N$$

for sufficiently large samples via the Laplace's method, where d is the number of free parameters and $\hat{\Theta}$ is the MLE under the model M. Using this approximation, Schwarz (1978) defined BIC as

$$\text{BIC} = -2\log P(Y|\hat{\Theta}, M) + d\log N,$$

which is an approximation of the negative of the log-posterior, and proposed to select the model which minimizes BIC.

Rissanen (1978) proposed the MDL principle, which seeks to minimize the number of bits needed to describe the data over the class of models. The total description length is defined by

$$l(Y, \Theta, M) = l(M) + l(\Theta|M) + l(Y|\Theta, M).$$

The first term of the above equation indicates the encoding length of the model M, the second encodes the parameters, and the third encodes the data Y given the model and parameters. The MDL of model selection chooses the model M and the parameters Θ which minimize $l(Y, \Theta, M)$ for the given data Y. Applying Shannon's theory, we can show the code length $l(Y|\Theta, M)$ for the data Y, equals the negative log likelihood $-\log P(Y|\Theta, M)$. In addition, if we accept the fact $p(\Theta|M) \propto e^{-l(\Theta|M)}$, the MDL principle can be interpreted as the maximization of the logarithm of the posterior probability of the model, which is essentially the same as minimizing BIC. Since Rissanen (1978) first proposed MDL, many modifications have been suggested and applied successively by Wallace and Freeman (1987), Quinlan and Rivest (1989), Mehta et al. (1995) and Takeuchi (1997).

2.3 Minimum Error Pruning(MEP)

Niblett and Bratko (1986) proposed a bottom-up approach seeking for a single tree that minimizes "the expected error rate on an independent data set." For a K-class problem, the expected probability that an observation at node t belongs to the i-th class is defined as

$$p_i(t) = \frac{n_i(t)}{n(t)} \frac{n(t)}{n(t)+m} + \pi_i \frac{m}{n(t)+m}$$

where $n(t)$ is the number of observations in node t, $n_i(t)$ is the number of observations in node t whose class level is i, π_i is the prior probability of the i-th class, and m is a tuning parameter that determines the impact of the prior probability. Cestnik and Bratko (1991) named $p_i(t)$ as m-probability estimate. Actually, $p_i(t)$'s are Bayes estimators when we assume that the distribution of class levels are multinomial, and the prior probabilities are distributed according to the Dirichlet distriution.

By letting $m = K$ and $\pi_i = 1/K$, $i = 1, \ldots, K$, i.e, the prior probabilities are equal for the all classes, Niblett and Bratko (1986) defined the expected error rate as

$$EER(t) = \min_i \{1 - p_i(t)\} = \min_i \left\{ \frac{n(t) - n_i(t) + K - 1}{n(t) + K} \right\}.$$

Using this measure, minimum error pruning is carried out as follows :

(1) Start with the full grown tree.
(2) For each internal node t,
- Compute the expected error : $EER(t)$, and
- Compute the error of the sub-tree given by : $EER(T_t) = \sum_{s \in T_t} p_s EER(s)$, where $p_s = n(s)/n(t)$ is the proportion of the number of samples in node s.
- A node t is pruned if $EER(t) \leq EER(T_t)$, or not otherwise.

3 Maximum a posteiori pruning algorithm

In this section, we propose a new pruning algorithm which combines the ideas of the CCP and Bayesian approach. First, we explain how to calculate the posterior probability of a given tree, and then explain how to use the posterior probabilities to select the best tree.

The structure of a given tree T is composed of the terminal nodes $\tilde{T}$ and parameters $\Theta = (\theta_1, \ldots, \theta_{\tilde{T}})$ at the terminal nodes. Let y_{ti} denote the i-th observation in the t-th terminal node, $t = 1, ..., |\tilde{T}|$, $i = 1, ..., n_t$, and let

$$Y \equiv (Y_1, ..., Y_{|\tilde{T}|})', \text{ where } Y_t \equiv (y_{t1}, ..., y_{tn_t})'.$$

For a classification tree, it is typically assumed that, conditionally on (Θ, T), y values within the terminal node are independent and identical distributed, and y values across terminal nodes are independent. In addition, since y_{ti} belongs to one of K categories (i.e. K class classification problem), we assume $f(y_{ti}|\theta_t)$ to be a multinomial distribution. Then the likelihood becomes

$$P(Y|\Theta, T) = \prod_{t=1}^{|\tilde{T}|} \prod_{i=1}^{n_t} f(y_{ti}|\theta_t) = \prod_{t=1}^{|\tilde{T}|} p_{t1}^{n_{t1}} \cdots p_{tK}^{n_{tK}},$$

where $\theta_t = (p_{t1}, ..., p_{tK}), p_{tk} \geq 0$, $\sum_k p_{tk} = 1$ and n_{tk} is the number of observations in node t belonging to the class k.

For a prior on T, we need to choose two priors $\pi(T)$ and $\pi(\Theta|T)$, one for the structure of T and the other for Θ given T. To specify $\pi(T)$, we mimic the prior developed by Chipman et al. (1998). They considered the node splitting probability and the selection probabilities of the split variables. However we use only the node splitting probability. The reason for this modification is as follows. The proposed pruning algorithm chooses the optimal tree among the sequence of nested trees already constructed from the growing step. Hence, the random quantity in the tree structure is the size of the tree, but not the split variables.

For a given node t, the split probability is defined by

$$\Pr(\textit{node split}) = \alpha(1 + d_t)^{-\beta}$$

where d_t is the depth of the node t and the parameters $\alpha \in (0, 1)$ and $\beta > 0$ control the size and shape of the tree. That is, the larger α is, the smaller the probability of the bushed tree is. On the other hand, for increasing β, it is difficult to split at deeper node. Using the above splitting probability, the total prior mass for a given tree T is specified by

$$\pi(T) = \prod_{t \in T - \tilde{T}} \alpha(1+d_t)^{-\beta} \prod_{t \in \tilde{T}} (1 - \alpha(1+d_t)^{-\beta}). \tag{2}$$

For $\pi(\Theta|T)$, we assume a priori $\pi(\Theta|T) = \prod_{t=1}^{|\tilde{T}|} \pi(\theta_t|T)$ and $\pi(\theta_t|T)$ are Dirichlet distributions with parameters $(\gamma_{t1}, ..., \gamma_{tK})$ which are conjugate to the multinomial distribution.

Once the likelihood and priors are specified, we can calculate the posterior probability of a tree T given data Y from the relation

$$P(T|Y) \propto P(Y|T)\pi(T), \tag{3}$$

where $P(Y|T) = \int P(Y|\Theta, T)\pi(\Theta|T)d\Theta$, which turns out to be

$$P(Y|T) = \prod_{t=1}^{|\tilde{T}|} \Big(\frac{\Gamma(\sum_k \gamma_k)}{\Gamma(n_t + \sum_k \gamma_k)} \prod_{k=1}^{K} \frac{\Gamma(n_{tk} + \gamma_k)}{\Gamma(\gamma_k)} \Big), \tag{4}$$

provided $\gamma_{tk} = \gamma_k$ for all t.

Now, we present the proposed pruning algorithm using the posterior probabilities (3) through the prior (2) and the marginal likelihood (4). For all subtrees of initial fully grown tree, we could evaluate the posterior probabilities and then select the maximum a posteriori(MAP) tree. However, the number of all subtrees evaluated is very large, which requires much computational resource. So, we need a reduced set of subtrees instead of all possible subtrees, and the CCP is an elegant algorithm for it.

The following pruning algorithm, called CCP-MAP, combines CCP and the maximum a posteriori strategy. That is, it first generates a sequence of nested trees using CCP, and then calculates the posteriori probabilities of the nested trees to select the optimal one.

Algorithm 1. MAP-pruning via cost-complexity pruned trees(CCP-MAP)

```
1   Grow the tree up to its maximum size;
2   Get the sequence of nested pruned trees $\{T_1, \ldots, T_n\}$ using CCP;
3   for $i = 1$ to $n$ do
        Compute the posterior probability $P(T_i|data)$ from the eq. (3);
    end for
4   Selection : $T^{MAP} = \arg\max\{P(T_i|data), i = 1, \ldots, n\}$;
```

4 BUMPing based on MAP-pruning

An important application of the proposed CCP-MAP pruning algorithm is the BUMPing procedure suggested by Tibshirani and Knight (1999). BUMPing is a method to select the best one among the set of trees constructed on bootstrap samples. In the problems where there are many local minima (or local maxima) such as decision trees, BUMPing can help avoiding getting stuck in a poor solution.

Let $\mathcal{L}$ be the training sample and $\mathcal{L}^b$ be a bootstrap sample obtained from $\mathcal{L}$. The standard algorithm of BUMPing is as follows.

(1) Let $R_\lambda(T) = R(T) + \lambda|\tilde{T}|$ be the cost-complexity measure for a given tree T where R is the misclassification error.
(2) Estimate the complexity parameter $\hat{\lambda} = \arg\min_\lambda R_\lambda(T)$ in the usual way from the training sample $\mathcal{L}$ by cross-validation.
(3) For each bootstrap samples $\mathcal{L}^b, b = 1, ..., B$, find the best tree $T^b = T_{\hat{\lambda}}(\mathcal{L}^b)$.
(4) Choose the tree $T^* = \arg\min R(T^b)$, where R is evaluated from the original training sample $\mathcal{L}$.

One problem of the standard BUMPing algorithm is that the optimal complexity parameter $\hat{\lambda}$ obtained from the original sample $\mathcal{L}$ may not be optimal for bootstrap samples. For this reason, BUMPing may yield sub-optimal results. To improve the performance of BUMPing, we can apply CCP-MAP to the BUMPing algorithm instead of the CV. This strategy makes it possible to compare larger number of trees than usual BUMPing, so that gives more chances of finding the optimal one. The proposed BUMPing algorithm based on CCP-MAP, called BUMP-MAP, is given as follows.

Algorithm 2. BUMPing based on CCP-MAP(BUMP-MAP)

```
1   for $b = 1$ to $B$ do
        Make a bootstrap sample $\mathcal{L}^b$ from $\mathcal{L}$;
        Grow tree $T^{(b)}$ from $\mathcal{L}^b$;
        Construct a sequence of pruned trees via CCP: $\{T_k^{(b)}, k = 1, \ldots, n_b\}$;
        Compute $\mathrm{P}(T_k^{(b)}|\mathcal{L})$ from the training data $\mathcal{L}$;
    end for
2   Choose the tree having the maximum a posteriori probability;
```

5 Empirical Results

5.1 Experimental setup

In this section, we compare the proposed method with other pruning algorithms by analyzing 10 real data sets listed in Table 1, which are available in UCI machine learning repository (http://www.ics.uci.edu/∼mlearn /ML-Repository.html). For evaluation of the performance, we first divide the data set randomly into a training set (70%) and test set (30%). This random division of the data set is repeated 20 times. All the following results are the summary over the 20 iterations.

For CCP-MAP pruning, we should determine the hyperparmeters α, β and γ's appropriately. We use a pair $(0.9, 1.0)$ for (α, β) that produces the best results empirically. For γ's, we choose 1's on two-class problems because the corresponding Dirichlet distribution becomes the uniform distribution which is thought to be a noninformative prior. However, in the case of multi-class problems, this choice results in poor accuracies since the total contribution of the prior to the posterior measured by $\sum_k \gamma_k$ is too large. Thus we set the total contribution to 2 as is the case for the two-class problem. That is, we set $\gamma_k = 2/K$ for K-class problems.

Table 1
The characteristics of the data sets used for the experiments

#	Data Sets	no. of obs.	no. of classes	no. of input vaiables
(1)	Australian	690	2	14
(2)	Breast Cancer	683	2	9
(3)	German	1,000	2	20
(4)	Glass	214	6	10
(5)	Ionosphere	351	2	34
(6)	Iris	150	3	3
(7)	Diabetes	768	2	8
(8)	Sonar	210	2	60
(9)	Vehicle	846	4	18
(10)	Vowel	999	11	10

5.2 Comparison of four pruning methods

We compare the four pruning methods - CCP-CV, CCP-MAP, CCP-BIC and MEP. Table 2 and 3 summarize the average generalization errors and sizes of

Table 2
The averaged misclassification error rates(%) and standard errors for the four pruning methods

	Pruning methods							
Data sets	CCP-CV		CCP-MAP		CCP-BIC		MEP	
Australian	15.12	(0.51)	15.05	(0.51)	14.98	(0.53)	15.82	(0.56)
Breast Cancer	5.07	(0.28)	5.14	(0.32)	5.02	(0.31)	5.17	(0.32)
German	27.67	(0.64)	28.70	(0.66)	29.13	(0.61)	30.90	(0.51)
Glass	34.69	(1.19)	35.23	(1.16)	35.86	(1.12)	34.92	(1.16)
Ionosphere	10.76	(0.72)	12.67	(0.61)	12.24	(0.65)	12.52	(0.55)
Iris	6.78	(0.65)	6.56	(0.67)	6.44	(0.60)	6.89	(0.70)
Diabetes	25.33	(0.40)	25.33	(0.50)	26.24	(0.56)	28.87	(0.60)
Sonar	26.05	(1.12)	25.08	(1.22)	25.65	(1.01)	25.40	(1.28)
Vehicle	30.33	(0.77)	30.31	(0.47)	30.04	(0.34)	29.72	(0.39)
Vowel	35.37	(0.77)	32.64	(0.75)	42.59	(0.92)	33.55	(0.66)

Table 3
The averaged size of trees and standard errors for the four pruning methods

	Pruning methods							
Data sets	CCP-CV		CCP-MAP		CCP-BIC		MEP	
Australian	12.15	(2.23)	14.30	(0.88)	17.05	(0.65)	34.85	(0.56)
Breast Cancer	5.50	(0.35)	10.75	(0.43)	8.45	(0.25)	12.10	(0.27)
German	10.10	(1.35)	24.00	(1.82)	44.65	(1.33)	73.70	(0.88)
Glass	10.65	(0.76)	9.80	(0.40)	6.65	(0.29)	13.10	(0.53)
Ionosphere	4.60	(0.50)	9.45	(0.30)	8.35	(0.39)	12.15	(0.27)
Iris	3.35	(0.17)	3.65	(0.15)	3.25	(0.10)	4.90	(0.16)
Diabetes	8.75	(1.63)	17.45	(1.02)	31.65	(1.17)	55.45	(0.55)
Sonar	6.65	(0.56)	10.20	(0.33)	9.20	(0.21)	11.55	(0.14)
Vehicle	27.80	(2.43)	21.40	(1.12)	31.40	(1.18)	57.70	(0.49)
Vowel	53.50	(2.69)	58.85	(0.74)	36.10	(0.97)	55.65	(0.88)

the final trees for 20 randomly partitioned data sets.

CCP-CV and CCP-MAP yield similar generalization errors. For tree sizes, the standard errors of CCP-CV are consistently larger than those of CCP-MAP, though the averaged tree sizes of CCP-CV tend to be smaller. This implies that CCP-MAP gives more stable results. We believe that the instability in CCP-CV is mainly due to the instability of the CV.

MEP tends to yield larger trees. In particular, for two data sets (`German` and `Diabetes`), the final trees of MEP are unnecessarily large. It seems that MEP could overfit the training data (i.e. lower accuracies). The results of BIC are data dependent. For `German` and `Diabetes`, the tree sizes are too large that the accuracies are worse than those of CCP-CV and CCP-MAP. On the other hand, for `Vowel`, the resulting tree size is too small to give the lowest accuracy.

5.3 Comparisons of the two bumping methods

We also compare the performance of the BUMPing procedures based on the CV and MAP. From Table 4, we can see that BUMP-MAP improves significantly standard BUMPing (BUMP-CV) in most cases (8 out of 10 data sets). For the two data sets `Vehicle` and `Vowel` where the accuracies of BUMP-MAP are lower that those of BUMP-CV, the tree sizes of BUMP-MAP are too small compared to BUMP-CV. However, note that the MAP pruning algorithm can control the sizes of trees by controlling the prior parameters (i.e. α and β

Table 4
The averaged tree sizes and error rates by the two BUMPing procedures

Data set	tree sizes(s.e)		average errors(s.e)	
	BUMP-CV	BUMP-MAP	BUMP-CV	BUMP-MAP
Australian	7.55 (1.00)	14.95 (0.74)	13.72 (0.48)	11.98 (0.51)
Breast Cancer	5.35 (0.41)	6.60 (0.21)	6.40 (0.38)	4.14 (0.25)
German	17.30 (3.02)	28.45 (0.82)	25.08 (0.72)	21.37 (0.65)
Glass	9.85 (0.70)	7.95 (0.38)	34.61 (1.47)	30.55 (0.98)
Ionosphere	5.25 (0.36)	7.15 (0.22)	12.19 (0.75)	8.67 (0.70)
Iris	3.35 (0.15)	3.00 (0.00)	6.22 (0.70)	5.56 (0.52)
Diabetes	14.35 (2.05)	22.70 (0.83)	23.83 (0.52)	20.30 (0.57)
Sonar	5.60 (0.41)	7.25 (0.27)	27.34 (1.57)	20.48 (0.85)
Vehicle	26.60 (1.13)	18.75 (0.53)	24.82 (0.62)	25.93 (0.57)
Vowel	48.30 (1.70)	27.40 (0.48)	33.57 (0.74)	39.46 (0.69)

in eq.(2)). By doing so, we can improve the performance of BUMP-MAP for these two data sets.

6 Discussions

BUMP-MAP can be compared with Bayesian CART proposed by Chipman et al. (1998). The common feature of the two algorithms is to search the maximum a posteriori tree. The difference is that the candidate trees are constructed by the MCMC algorithm in Bayesian CART while BUMP-MAP generates them using bootstrap samples and applying the CCP algorithm. One disadvantage of Bayesian CART is that the MCMC algorithm requires too much computation so that it cannot be applied to large data sets easily. It is clear that the proposed method needs much less computation.

Instead of choosing the best one from many trees in BUMP-MAP, we can combine all trees into one final model similarly to bagging (Breiman, 1996). While all trees are averaged out in bagging, we can use the weighted average of trees with weights proportional to their posterior probabilities. That is,

$$\tilde{T}(\mathbf{x}) = \sum_{i=1} w_i T_i(\mathbf{x}),$$

where $w_i = \frac{P(T_i|data)}{\sum_j P(T_j|data)}$. This approach can be considered as an approximation of Bayesian model averaging. We will pursue this idea in future.

Acknowledgements

The authors would like to thank the associate editor and anonymous referees for their helpful comments. This research was supported in part by US Air Force Research Grant F62562-03-P-0658 and in part by KOSEF through Statistical Research Center for Complex Systems at Seoul National University, Korea.

References

Breiman, L., 1996. Bagging predictors. Machine Learning 24 (2), 123–140.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth.

Cestnik, B., Bratko, I., 1991. On estimating probabilities in tree pruning. In: Proceedings of European Working Sessions on Learning EWSL 91, Lecture notes in artificial intelligence. Vol. 482. Springer-Verlag, pp. 138–150.

Chipman, H., George, E., McCulloch, R., 1998. Bayesian CART model search. J. Am. Statist. Assoc. 93, 937–960.

Mehta, M., Rissanen, J., Agrawal, R., 1995. MDL-based decision tree pruning. In: Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95). pp. 216–221.

Niblett, T., Bratko, I., 1986. Learning decision rules in noisy domains. In: Research and Development in Expert Systems III. Cambridge University Press, Cambridge, pp. 25–34.

Quinlan, J. R., 1993. C4.5 : Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA.

Quinlan, J. R., Rivest, R. L., 1989. Inferring decision trees using the minimum description length principle. Information and Computation 80, 227–248.

Rissanen, J., 1978. Modeling by shortest data description. Automatica 14, 465–471.

Schwarz, G., 1978. Estimating the dimension of a model. The Annals of Statistics 6, 461–464.

Takeuchi, J., 1997. Characterization of the Bayes estimator and the MDL estimator for exponential families. IEEE Transactions on Information Theory 43 (4), 1165–1174.

Tibshirani, R., Knight, K., 1999. Model search and inference by bootstrap "bumping". Journal of the Computational and Graphical Statistics, 671–686.

Wallace, C. S., Freeman, P., 1987. Estimation and inferance by compact coding. Journal of the Royal Statistical Society **B** 42 (3), 241–252.

A gradient descent algorithm for generalized LASSO

Jinseog Kim JSKIM@STATS.SNU.AC.KR
Statistical Research Center for Complex Systems
Seoul National University, Seoul 151-742, Korea

Yuwon Kim GARY@STATS.SNU.AC.KR
Statistical Research Center for Complex Systems
Seoul National University, Seoul 151-742, Korea

Yongdai Kim YDKIM@STATS.SNU.AC.KR
Department of Statistics
Seoul National University,
Seoul 151-742, Korea

Editor: Leslie Pack Kaelbling

Abstract

LASSO is an useful method to achieve the shrinkage and variable selection simultaneously. The main idea of LASSO is to use the L_1 constraint in the regularization step. Starting from linear models, the idea of LASSO - using the L_1 constraint, has been applied to various models such as wavelets, kernel machines, smoothing splines, multiclass logistic models etc. These models extend the standard LASSO in two ways - there is more than one L_1 constraint in the regularization step and there is more than one linear combination of inputs in the model. We call such models the *generalized LASSO* models. In this paper, we propose a new computational algorithm for the generalized LASSO, which is computationally much simpler than standard algorithms such as QP or non-linear program. We also prove that the convergence speed is fast and does not depend on the dimension of inputs. Simulations show that the proposed algorithm is fairly fast and provides reliable results. To illustrate its computing power with high dimensional data, we analyze real data sets using the structural sparse kernel machine and multiclass logistic regression model.

Keywords: Gradient descent, LASSO, Multiclass logistic model, Sparse kernel machines, Variable selection

1. Introduction

Tibshirani (1996) introduced an interesting method for shrinkage and variable selection in linear models, called "Least Absolute Shrinkage and Selection Operator (LASSO)". It achieves better prediction accuracy by shrinkage as the ridge regression, but at the same time, it gives a sparse solution, which means that some coefficients are exactly 0. Hence, LASSO can be thought to achieve the shrinkage and variable selection simultaneously.

The main idea of LASSO is to use the L_1 constraint in the regularization step. That is, the estimator is obtained by minimizing the empirical risk subject to that the L_1 norm of the regression coefficients is bounded by a given positive numbers. Starting from linear models, the idea of LASSO - using the L_1 constraint, has been applied to various models such as wavelets (Chen et al., 1999; Bakin, 1999), kernel machines (Gunn and Kandola, 2002; Roth, 2004), smoothing splines (Zhang et al., 2003), multiclass logistic regressions (Krishnapuram et al., 2004) etc. These models extend the standard LASSO in two ways - there is more than one L_1 constraint in the regularization step (Gunn and Kandola, 2002; Zhang et al., 2003) and there is more than one linear combination of inputs in the model such as the sparse multiclass logistic model (Krishnapuram et al., 2004). We call such models the *generalized LASSO* models, whose details are given in section 2.

One important issue in the generalized LASSO is that the objective function is not differentiable due to the L_1 constraint, and hence special optimization techniques are necessary. Tibshirani (1996) used the quadratic programming (QP) method for least squares regressions and the iteratively reweighted least squares procedure (IRLS) with QP for logistic regressions. Osborne et al. (2000) proposed a faster QP algorithm for linear regressions, which was extended for logistic regressions and implemented by Lokhorst et al. (1999) as `lasso2` package in R system. Recently, Efron et al. (2004) developed an algorithm called LARS for linear regressions, which is shown to be closely related to Osborne et al. (2000)'s algorithm. The aforementioned algorithms, however, may not be easily applicable to large data sets when the dimension of input variables (or features) is very large. One such example is the structural sparse kernel machine (see Example 3 in Section 2), where the dimension of inputs is proportional to the sample size. This computational drawback is mainly due to that these algorithms need repeated matrix inversions whose computing cost increases very fast as the dimension of inputs increases. Also, these algorithms are not directly applicable to the generalized LASSO models where there are more than one L_1 constraint and/or there are more than one linear combination of inputs. There are some attempts to resolve these computational issues. Grandvalet and Canu (1999) implemented a fixed point algorithm using the equivalence between the adaptive ridge regression and LASSO, and Perkins et al. (2003) developed a stagewise gradient descent algorithm called GRAFTING in the more general set-up of LASSO. These algorithms, however, may not lead global convergence.

In this paper, we propose a gradient descent algorithm for the generalized LASSO. The proposed algorithm does not require matrix inversions, so that it can be applied to data sets with large dimensional inputs. Moreover, the algorithm always converges to the global optimum, and its convergence speed is near optimum among sequentially updating algorithms.

The paper is organized as follows. In Section 2, we define the general set-up of the generalized LASSO and give four examples. In Section 3, we propose a gradient descent algorithm for the generalized LASSO, and study its theoretical properties in Section 4. Section 5 carries out to compare the proposed algorithm with Osborne et al. (2000)'s QP algorithm in the standard LASSO framework. The proposed algorithm is applied to a structural sparse kernel machine and a multiclass logistic regression in Section 6 and Section 7, respectively. Concluding remarks follow in Section 8.

2. The set-up of generalized LASSO

We first present the set-up of the generalized LASSO and give four examples. Let $(y_1, \mathbf{x}_1)$, $\ldots$, $(y_n, \mathbf{x}_n)$ be n output/input pairs where $y_i \in \mathcal{Y}$ and $\mathbf{x}_i \in \mathcal{X}$ where $\mathcal{X}$ is a subspace of R^p, the p-dimensional Euclidean space. That is, $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$. Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$ be the corresponding regression coefficients. For a given loss function $\mathcal{L}$, the objective of the generalized LASSO is to find $\boldsymbol{\beta}$ which minimizes the (empirical) risk

$$R(\boldsymbol{\beta}) = \sum_{i=1}^{n} \mathcal{L}\left(y_i, \boldsymbol{\beta} \otimes \mathbf{x}_i\right) \tag{1}$$

subject to $||\boldsymbol{\beta}^l||_1 \leq \lambda_l$ for $l = 1, \ldots, d$ where $\boldsymbol{\beta}^l = (\beta_1^l, \ldots, \beta_{p_l}^l)$ are p_l dimensional subvectors of $\boldsymbol{\beta}$ for $l = 1, \ldots, d$ such that $\boldsymbol{\beta} = (\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^d)$. Here λ_ls are tuning parameters and $||\boldsymbol{\beta}^l||_1 = \sum_{k=1}^{p_l} |\beta_k^l|$. In (1), $\boldsymbol{\beta} \otimes \mathbf{x}_i = (\boldsymbol{\beta}^1 \otimes \mathbf{x}_i^1, \ldots, \boldsymbol{\beta}^d \otimes \mathbf{x}_i^d)$ and $\boldsymbol{\beta}^l \otimes \mathbf{x}_i^l = (\beta_1^l x_{i1}^l, \ldots, \beta_{p_1}^l x_{ip_l}^l)$ where $\mathbf{x}_i^l = (x_{i1}^l, \ldots, x_{ip_l}^l)$ are p_l dimensional subvectors of $\mathbf{x}_i$ for $l = 1, \ldots, d$ such that $\mathbf{x}_i = (\mathbf{x}_i^1, \ldots, \mathbf{x}_i^d)$.

Example 1. *Multiple linear regression*
Let $(y_1, \mathbf{z}_1), \ldots, (y_n, \mathbf{z}_n)$ be n observations where y_is are outputs and $\mathbf{z}_i$s are p-dimensional inputs. A multiple linear regression model is given by

$$y_i = \beta_1 z_{i1} + \cdots + \beta_p z_{ip} + \epsilon_i \tag{2}$$

where ϵ_is are assumed to be zero-mean random quantities. The LASSO estimate $\hat{\beta}_1, \ldots, \hat{\beta}_p$ for the linear regression model (2) is the minimizer of

$$\sum_{i=1}^{n} \left(y_i - \sum_{k=1}^{p} \beta_k z_{ik} \right)^2$$

subject to $\sum_{k=1}^{p} |\beta_k| \leq \lambda$. This problem can be embedded into the set-up of the generalized LASSO with $d = 1$, the loss function $\mathcal{L}(y, \mathbf{v}) = (y - \sum_{k=1}^{p} v_k)^2$ for $\mathbf{v} = (v_1, \ldots, v_p)$, $\mathbf{x}_i = \mathbf{z}_i$, $\boldsymbol{\beta} = \boldsymbol{\beta}^1 = (\beta_1, \ldots, \beta_p)$ and $\lambda_1 = \lambda$.

Remark. The model (2) is not of standard since there is no intercept term. The standard multiple linear regression model is

$$y_i = \beta_0 + \beta_1 z_{i1} + \cdots + \beta_p z_{ip} + \epsilon_i. \tag{3}$$

The LASSO estimates ($\hat{\beta}_0$ and $\hat{\beta}_1, \ldots, \hat{\beta}_p$) then can be obtained by minimizing

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k=1}^{p} \beta_k z_{ki} \right)^2$$

subject to $\sum_{k=1}^{p} |\beta_k| \leq \lambda$. Note that β_0 is unconstrained. This LASSO problem can be embedded into the generalized LASSO by letting $d = 2$, $\mathbf{x}_i = (1, \mathbf{z}_i)$ and $\boldsymbol{\beta}^1 = (\beta_0), \boldsymbol{\beta}^2 = (\beta_1, \ldots, \beta_p)$ and $\lambda_1 = \infty, \lambda_2 = \lambda$. In practice, we can use a sufficient large number for λ_1 such that $\hat{\beta}_0^{\mathrm{OLS}} \leq \lambda_1$ where $\hat{\beta}_0^{\mathrm{OLS}}$ is the OLS estimate of β_0. To sum up, the models with (unconstrained) intercept terms can be easily accommodated with the generalized LASSO set-up without intercept terms. Hence, for ease of notation, in the followings, we only consider the models without intercept terms.

Example 2. *Multiple logistic regression*
A multiple logistic regression model for two class classification problems is given by

$$\log \frac{\Pr(y_i = 1|\mathbf{z}_i)}{1 - \Pr(y_i = 1|\mathbf{z}_i)} = f(\mathbf{z}_i) \tag{4}$$

where

$$f(\mathbf{z}_i) = \beta_1 z_{i1} + \cdots + \beta_p z_{ip}.$$

Here, $y_i \in \{0, 1\}$. The LASSO estimate $\hat{\beta}_1, \ldots, \hat{\beta}_p$ is the minimizer of the negative binomial log likelihood

$$\sum_{i=1}^{n} \left[-y_i f(\mathbf{z}_i) + \log \left(1 + e^{f(\mathbf{z}_i)} \right) \right]$$

subject to $\sum_{k=1}^{p} |\beta_k| \leq \lambda$. The generalized LASSO setting of a multiple logistic regression is the same as that of the multiple linear regression except for the loss function

$$\mathcal{L}(y, \mathbf{v}) = -y \left(\sum_{k=1}^{p} v_k \right) + \log \left\{ 1 + \exp \left(\sum_{k=1}^{p} v_k \right) \right\}.$$

Example 3. *Structural sparse kernel machine*
The structural sparse kernel machine with second order interaction terms for classification is basically the same as (4) except that $f(\mathbf{z})$ is modelled via the functional ANOVA (analysis of variance) decomposition

$$f(\mathbf{z}) = \sum_{j=1}^{p} f_j(z_j) + \sum_{j<k} f_{jk}(z_j, z_k). \tag{5}$$

Then, we assume that f_j and f_{jk} are elements of the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_j$ and $\mathcal{H}_{jk}$ with the reproducing kernels K_j and K_{jk}, respectively. Motivated by Kimeldorf and Wahba (1972), we assume that

$$f_j(z_j) = \sum_{r=1}^{n} c_{j,r} K_j(z_{rj}, z_j)$$

and

$$f_{jk}(z_j, z_k) = \sum_{r=1}^{n} c_{jk,r} K_{jk}((z_{rj}, z_{rk}), (z_j, z_k)).$$

This model is considered by many authors including Gunn and Kandola (2002) and Zhang et al. (2003). Zhang et al. (2003) proposed to estimate $c_{j,r}$ and $c_{jk,r}$ by minimizing

$$\sum_{i=1}^{n} \mathcal{L}(y_i, f(\mathbf{z}_i))$$

subject to $\sum_{r=1}^{n} \sum_{j=1}^{p} |c_{j,r}| \leq \lambda_1$ and $\sum_{r=1}^{n} \sum_{j<k} |c_{jk,r}| \leq \lambda_2$. This problem can be embedded into the setting of the generalized LASSO with $d = 2$, $\mathbf{x}_i = (\mathbf{x}_i^1, \mathbf{x}_i^2)$ and $\boldsymbol{\beta} = (\boldsymbol{\beta}^1, \boldsymbol{\beta}^2)$

where $\mathbf{x}_i^1 = (K_j(z_{rj}, z_{ij}), j = 1, \ldots, p, r = 1, \ldots, n)$ and $\mathbf{x}_i^2 = (K_{jk}((z_{rj}, z_{rk}), (z_{ij}, z_{ik})), j < k, r = 1, \ldots, n)$, $\boldsymbol{\beta}^1 = (c_{j,r}, j = 1, \ldots, p, r = 1, \ldots, n)$, and $\boldsymbol{\beta}^2 = (c_{jk,r}, j < k, r = 1, \ldots, n)$. Note that the number of parameters to be estimated is proportional to the sample size n. Hence, the dimension of inputs can be very large when the sample size is large even though the dimension of the original inputs (i.e. p) is small.

Example 4. *Multiclass logistic regression model*
A J-class logistic regression model is given by

$$\Pr(y_i = j|\mathbf{z}_i) = \frac{\exp\left(\beta_1^{(j)} z_{i1}^{(j)} + \cdots + \beta_p^{(j)} z_{ip}^{(j)}\right)}{\sum_{m=1}^J \exp\left(\beta_1^{(m)} z_{i1}^{(m)} + \cdots + \beta_p^{(m)} z_{ip}^{(m)}\right)}$$

for $j = 1, \ldots, J$. For identification of the model, we let $\beta_k^{(J)} = 0$ for $k = 1, \ldots, p$. For a sparse solution, we estimate $\beta_k^{(j)}$s by maximizing the log-likelihood

$$\sum_{i=1}^n \left[\sum_{j=1}^J I(y_i = j) f_j(\mathbf{z}_i^{(j)}) - \log\left(\sum_{m=1}^J \exp\left(f_m\left(\mathbf{z}_i^{(m)}\right)\right)\right)\right]$$

subject to certain L_1 constraints on $\beta_k^{(j)}$s where $\mathbf{z}_i^{(j)} = (z_{i1}^{(j)}, \ldots, z_{ip}^{(j)})$ and

$$f_j(\mathbf{z}_i^{(j)}) = \beta_1^{(j)} z_{i1}^{(j)} + \cdots + \beta_p^{(j)} z_{ip}^{(j)}$$

for $j = 1, \ldots, J-1$ and $f_J(\mathbf{z}_i^{(J)}) = 0$. If we put the constraint $\sum_{j=1}^{J-1} \sum_{k=1}^p |\beta_k^{(j)}| \le \lambda$ (Krishnapuram et al., 2004), this is a generalized LASSO model with $d = 1$, $\mathbf{x}_i = (\mathbf{z}_i^{(1)}, \ldots, \mathbf{z}_i^{(J)})$, $\boldsymbol{\beta} = (\beta_k^{(j)}, k = 1, \ldots, p, j = 1, \ldots, J-1)$ and the loss function $\mathcal{L}$ given by

$$\mathcal{L}(y, \mathbf{v}) = -\sum_{j=1}^J I(y = j) \sum_{k=1}^p v_k^j + \log\left\{\sum_{j=1}^{J-1} \exp\left(\sum_{k=1}^p v_k^j\right) + 1\right\} \tag{6}$$

where $\mathbf{v} = (\mathbf{v}^1, \ldots, \mathbf{v}^{J-1})$ and $\mathbf{v}^j = (v_1^j, \ldots, v_p^j)$.

In some cases, we want to put a different L_1 constraint on each $\boldsymbol{\beta}^{(j)} = (\beta_1^{(j)}, \ldots, \beta_p^{(j)})$, that is $\sum_{k=1}^p |\beta_k^{(j)}| \le \lambda_j, j = 1, \ldots, J-1$. This is also a generalized LASSO model with $d = J-1$, $\mathbf{x}_i = (\mathbf{z}_i^1, \ldots, \mathbf{z}_i^{J-1})$, and $\boldsymbol{\beta} = (\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^{J-1})$ and the loss function $\mathcal{L}$ given in (6).

3. A gradient descent algorithm for generalized LASSO

In this section, we will present a gradient descent algorithm to solve the optimization problem in the generalized LASSO. Let $\mathbf{w}^l = \boldsymbol{\beta}^l/\lambda_l$ and let $\tilde{\mathbf{x}}_i = (\tilde{\mathbf{x}}_i^1, \ldots, \tilde{\mathbf{x}}_i^d)$ where $\tilde{\mathbf{x}}_i^l = \lambda_l \mathbf{x}_i^l$. Then, we can change (1) to the equivalent problem which finds $\hat{\mathbf{w}} = (\hat{\mathbf{w}}^1, \ldots, \hat{\mathbf{w}}^d)$ by minimizing

$$R(\mathbf{w}) = \sum_{i=1}^n \mathcal{L}\left(y_i, \mathbf{w} \otimes \tilde{\mathbf{x}}_i\right), \text{ subject to } ||\mathbf{w}^l||_1 \le 1,\ l = 1, \ldots, d. \tag{7}$$

To describe the proposed gradient descent algorithm, we introduce some notations. Let $\mathbf{z}_k^l = (\tilde{x}_{k1}^l, \dots, \tilde{x}_{kn}^l)'$ and let

$$\mathcal{S}^l = \{(w_1^l \mathbf{v}_l^l, \dots, w_{p_l}^l \mathbf{v}_{p_l}^l) : \mathbf{v}_k^l \in \{\mathbf{z}_k^l, -\mathbf{z}_k^l\}, w_k^l \geq 0, \sum_{k=1}^{p_l} w_k^l = 1\}.$$

That is, $\mathcal{S}^l$ is the set of $n \times p_l$ dimensional matrices whose column vectors are $w_k^l \mathbf{v}_k^l$ for $k = 1, \dots, p_l$. Finally, for given $n \times p_l$ dimensional matrices $\mathbf{F}^l \in \mathcal{S}^l$ for $l = 1, \dots, d$, let $\mathbf{F}$ be a $n \times p$ dimensional matrix defined by $\mathbf{F} = (\mathbf{F}^1, \dots, \mathbf{F}^d)$ and let $\mathcal{S} = \{\mathbf{F} : \mathbf{F}^l \in \mathcal{S}^l, l = 1, \dots, d\}$.

For a given matrix $\mathbf{F}$, define the cost function $C : \mathcal{S} \to R$ by

$$C(\mathbf{F}) = \sum_{i=1}^{n} \mathcal{L}(y_i, \mathbf{u}_i)$$

where $\mathbf{u}_i$ are the row vectors of $\mathbf{F}$. Then, finding $\hat{\mathbf{w}}$ which minimizes (7) is equivalent to find $\hat{\mathbf{F}}$ in $\mathcal{S}$ such that

$$\hat{\mathbf{F}} = \arg \min_{\mathbf{F} \in \mathcal{S}} C(\mathbf{F}). \tag{8}$$

The main idea of the proposed algorithm is to find $\hat{\mathbf{F}}$ sequentially as follows. Rewrite $\mathbf{F}^l = (\mathbf{f}_1^l, \dots, \mathbf{f}_{p_l}^l)$ where $\mathbf{f}_k^l$s are n-dimensional column vectors. For a given $\mathbf{V}^l \in \mathcal{S}^l$, let $\mathbf{F}[\alpha, \mathbf{V}^l] = (\mathbf{F}^1, \dots, \mathbf{F}^{l-1}, \mathbf{F}^l + \alpha(\mathbf{V}^l - \mathbf{F}^l), \mathbf{F}^{l+1}, \dots, \mathbf{F}^d)$. Suppose that $\mathbf{F}^l$ is the current estimate of $\hat{\mathbf{F}}^l$. Then for each l, the proposed algorithm searches a direction matrix $\mathbf{V}^l \in \mathcal{S}^l$ such that $C(\mathbf{F}[\alpha, \mathbf{V}^l])$ decreases most rapidly for some $\alpha \in [0, 1]$ and updates $\mathbf{F}$ to $\mathbf{F}[\alpha, \mathbf{V}^l]$. Note that $\mathbf{F}[\alpha, \mathbf{V}^l]$ is still in $\mathcal{S}$. Now, the Taylor expansion implies

$$C(\mathbf{F}[\alpha, \mathbf{V}^l]) \approx C(\mathbf{F}) + \alpha \sum_{k=1}^{p_l} \langle \nabla C(\mathbf{F})_k^l, \mathbf{v}_k^l - \mathbf{f}_k^l \rangle,$$

where $\nabla C(\mathbf{F})_k^l = \partial C(\mathbf{F}) / \partial \mathbf{f}_k^l$. Here, $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^n a_i b_i$ for $\mathbf{a}, \mathbf{b} \in R^n$. Moreover, it can be easily shown that

$$\min_{\mathbf{V}^l \in \mathcal{S}^l} \sum_{k=1}^{p_l} \langle \nabla C(\mathbf{F})_k^l, \mathbf{v}_k^l - \mathbf{f}_k^l \rangle = \min_{k=1,\dots,p_l} \min \left\{ \langle \nabla C(\mathbf{F})_k^l, \mathbf{z}_k^l - \mathbf{f}_k^l \rangle, \langle \nabla C(\mathbf{F})_k^l, -\mathbf{z}_k^l - \mathbf{f}_k^l \rangle \right\}. \tag{9}$$

Hence, the desired direction matrix $\mathbf{V}^l$ is such that the k-th column vector is either $\mathbf{z}_k^l$ or $-\mathbf{z}_k^l$ whichever minimizes the right hand side of (9) and the other column vectors are $\mathbf{0}$. By summing up the above arguments, we propose the gradient descent algorithm for the generalized LASSO in Figure 1.

The proposed algorithm is equivalent to the feasible direction method (see, for example, Bertsekas (1999)) for the standard LASSO model. In the standard LASSO model, $d = 1$ and the cost function $C(\mathbf{F})$ is a function of $\sum_{k=1}^p \mathbf{f}_k$. Then, to find the optimal $\hat{\mathbf{F}}$ in (8) is equivalent to find the optimal vector $\hat{\mathbf{h}}$ defined by

$$\hat{\mathbf{h}} = \text{argmin}_{\mathbf{h} \in \mathcal{S}} C(\mathbf{h})$$

```
1. Initialize : $\mathbf{w}^l = \mathbf{0}$, $\mathbf{F}_0^l = \mathbf{0}$ for $l = 1, \ldots, d$ and $\mathbf{F}_0 = (\mathbf{F}_0^1, \ldots, \mathbf{F}_0^d)$.
2. For $m = 1, \ldots, M$ do ;
    (a) Compute the gradient $\nabla C(\mathbf{F}_{m-1})$
    (b) Find the $(\hat{l}, \hat{k}, \hat{\gamma})$ which minimizes inner products
            $$\phi_m(l,k,\gamma) = \langle \nabla C(\mathbf{F}_{m-1})_k^l, \gamma \mathbf{z}_k^l \rangle,\ l = 1, \ldots, d, k = 1, \ldots, p_l, \gamma = \pm 1,$$
        and let $\mathbf{V}^{\hat{l}}$ be the $n \times p_{\hat{l}}$ matrix such that the $\hat{k}$-th column vector of $\mathbf{V}^{\hat{l}}$ is $\hat{\gamma}\mathbf{z}_{\hat{k}}^{\hat{l}}$ and the other column vectors are $\mathbf{0}$.
    (c) If $\phi_m(\hat{l}, \hat{k}, \hat{\gamma}) = 0$, then return $\mathbf{w} = (\mathbf{w}^l, \ldots, \mathbf{w}^d)$.
    (d) Else, find $\hat{\alpha} = \arg\min_{\alpha \in [0,1]} C\left(\mathbf{F}_{m-1}[\alpha, \mathbf{V}^{\hat{l}}]\right)$.
    (e) Update $\mathbf{F}$ and $\mathbf{w}$:
            $$\begin{aligned} \mathbf{F}_m &= \mathbf{F}_{m-1}[\hat{\alpha}, \mathbf{V}^{\hat{l}}] \\ w_k^l &= \begin{cases} (1-\hat{\alpha})w_k^l + \hat{\gamma}\hat{\alpha} & , l = \hat{l}, k = \hat{k} \\ (1-\hat{\alpha})w_k^l & , l = \hat{l}, k \neq \hat{k} \\ w_k^l & , o.w. \end{cases} \end{aligned}$$
3. Return $\mathbf{w} = (\mathbf{w}^l, \ldots, \mathbf{w}^d)$.
```

Figure 1: Gradient descent algorithm for generalized LASSO

where $\mathcal{S} = \{\sum_{k=1}^p w_k \mathbf{v}_k, \mathbf{v}_k \in \{\mathbf{z}_k, -\mathbf{z}_k\}, w_k \geq 0, \sum_{k=1}^p w_k = 1\}$. Note that $\mathcal{S}$ is the convex hull of $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_p\} \cup \{-\mathbf{z}_1, \ldots, -\mathbf{z}_p\}$. That is, the problem of finding $\hat{\mathbf{h}}$ is an optimization problem over a convex set. One of the standard methods for the optimization over a convex set is the feasible direction method. In the feasible method, for given current solution $\mathbf{h}_m$, we find $\hat{\mathbf{v}} \in \mathcal{S}$ which minimizes $\langle \nabla C(\mathbf{h}_m), \mathbf{v} \rangle$ on $\mathbf{v} \in \mathcal{S}$. Then, we find $\hat{\alpha}$ which minimizes $C(\mathbf{h}_m + \alpha(\hat{\mathbf{v}} - \mathbf{h}_m))$ on $\alpha \in [0,1]$. Finally, we update $\mathbf{h}_m$ by $\mathbf{h}_{m+1} = \mathbf{h}_m + \hat{\alpha}(\hat{\mathbf{v}} - \mathbf{h}_m)$. Since $\text{argmin}_{\mathbf{v} \in \mathcal{S}} \langle \nabla C(\mathbf{h}_m), \mathbf{v} \rangle = \text{argmin}_{\mathbf{v} \in \mathcal{Z}} \langle \nabla C(\mathbf{h}_m), \mathbf{v} \rangle$, the feasible direction method for the standard LASSO model is the same as the proposed algorithm.

4. Convergence Analysis

We assume that C is convex and satisfies the Lipschitz with the Lipschitz constant L on $\mathcal{S}$. That is, for any two matrices whose k-th column vectors are $\mathbf{g}_k = (g_{1k}, \ldots, g_{nk})'$ and $\mathbf{h}_k = (h_{1k}, \ldots, h_{nk})'$ respectively,

$$\|\nabla C(\mathbf{G}) - \nabla C(\mathbf{H})\| \leq L\|\mathbf{G} - \mathbf{H}\|$$

where $\nabla C(\mathbf{G}) = \partial C(\mathbf{G})/\partial \mathbf{G}$, $\|\mathbf{G} - \mathbf{H}\|^2 = \sum_{k=1}^p \sum_{i=1}^n (g_{ik} - h_{ik})^2$. We assume that $\mathcal{S}$ is a bounded set. Let $M_1 = L \sup_{\mathbf{G},\mathbf{H} \in \mathcal{S}} \sup_{k=1,\ldots,p} \sum_{i=1}^n (g_{ik} - h_{ik})^2$. Let $C^* = \inf_{\mathbf{F} \in \mathcal{S}} C(\mathbf{F})$ and let $\Delta C(\mathbf{F}) = C(\mathbf{F}) - C^*$. Let $M_2 = \sup_{\mathbf{F} \in \mathcal{S}} \Delta C(\mathbf{F})$, and let $M = \max\{M_1, M_2\}$. In addition, for notational convenience, we denote that $\langle \mathbf{G}, \mathbf{H} \rangle = \sum_{k=1}^{p} \langle \mathbf{g}_k, \mathbf{h}_k \rangle$. The following Theorem is the main result of this section.

Theorem 1

$$\Delta C\left(\mathbf{F}_m\right) \leq \frac{2d^2 M}{m}. \tag{10}$$

First of all, we start with the following two lemmas before we prove theorem 1.

Lemma 2 *For any $\mathbf{F} \in \mathcal{S}$, $\mathbf{V}^l \in S^l$ and $\alpha \in [0,1]$,*

$$C\left(\mathbf{F}[\alpha, \mathbf{V}^l]\right) \leq C(\mathbf{F}) + \alpha \langle \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle + \frac{M\alpha^2}{2}$$

where $\nabla C(\mathbf{F})^l = \partial C(\mathbf{F}) / \partial \mathbf{F}^l$.

Proof. Define $\Phi : R \to R$ by $\Phi(\alpha) = C(\mathbf{F}[\alpha, \mathbf{V}^l])$. Let $\Phi^{'}(\alpha) = d\Phi(\alpha)/d\alpha$. Then, we have

$$\Phi^{'}(\alpha) = \langle \nabla C(\mathbf{F}[\alpha, \mathbf{V}^l])^l, \mathbf{V}^l - \mathbf{F}^l \rangle.$$

Hence,

$$\begin{aligned} |\Phi^{'}(\alpha) - \Phi^{'}(0)| &= \left| \langle \nabla C(\mathbf{F}[\alpha, \mathbf{V}^l])^l - \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle \right| \\ &\leq \|\nabla C(\mathbf{F}[\alpha, \mathbf{V}^l])^l - \nabla C(\mathbf{F})^l\| \|\mathbf{V}^l - \mathbf{F}^l\| \\ &\leq L\alpha \|\mathbf{V}^l - \mathbf{F}^l\|^2 \end{aligned}$$

for $\alpha \in [0,1]$. The first inequality is by Cauchy-Schwarz inequality. Thus,

$$\begin{aligned} \Phi^{'}(\alpha) &\leq \Phi^{'}(0) + L\alpha \|\mathbf{V}^l - \mathbf{F}^l\|^2 \\ &= \langle \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle + L\alpha \|\mathbf{V}^l - \mathbf{F}^l\|^2. \end{aligned}$$

Hence,

$$\begin{aligned} \Phi(\alpha) - \Phi(0) &= \int_0^\alpha \Phi^{'}(s) ds \\ &\leq \int_0^\alpha \left(\langle \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle + Ls \|\mathbf{V}^l - \mathbf{F}^l\|^2 \right) ds \\ &= \alpha \langle \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle + \frac{L}{2} \|\mathbf{V}^l - \mathbf{F}^l\|^2 \alpha^2. \end{aligned}$$

Since $\mathbf{V}^l$ and $\mathbf{F}^l$ are in $\mathcal{S}^l$, $L\|\mathbf{V}^l - \mathbf{F}^l\|^2 \leq M_1 \leq M$, which completes the proof. ■

Lemma 3 *For any given $\mathbf{F} \in \mathcal{S}$, let $\mathbf{V}^l$ be a matrix in $\mathcal{S}^l$ which minimizes $\langle \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle$ and let $\hat{l} = \arg\min_{l=1,\ldots,d} \{\langle \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle\}$. Let*

$$\hat{\alpha} = \arg \min_{\alpha \in [0,1]} C\left(\mathbf{F}[\alpha, \mathbf{V}^{\hat{l}}]\right).$$

Then,

$$\Delta C\left(\mathbf{F}[\hat{\alpha}, \mathbf{V}^{\hat{l}}]\right) \leq \Delta C(\mathbf{F}) - \frac{\Delta C(\mathbf{F})^2}{2d^2 M}.$$

Proof. For given $\mathbf{F} \in \mathcal{S}$, define the Bregman divergence $\mathrm{B}(\mathbf{W}, \mathbf{F})$ on $\mathbf{W} \in \mathcal{S}$ by

$$\mathrm{B}(\mathbf{W}, \mathbf{F}) = C(\mathbf{W}) - C(\mathbf{F}) - \langle \nabla C(\mathbf{F}), \mathbf{W} - \mathbf{F} \rangle. \tag{11}$$

Note that $\inf_{\mathbf{W} \in \mathcal{S}} \mathrm{B}(\mathbf{W}, \mathbf{F}) \geq 0$ for all $\mathbf{F} \in \mathcal{S}$ since C is convex.

Now, let $\mathbf{W}$ be a matrix in $\mathcal{S}$. Since

$$\langle \nabla C(\mathbf{F}), \mathbf{W} - \mathbf{F} \rangle = \sum_{l=1}^{d} \langle \nabla C(\mathbf{F})^l, \mathbf{W}^l - \mathbf{F}^l \rangle,$$

we have

$$\langle \nabla C(\mathbf{F}), \mathbf{W} - \mathbf{F} \rangle \geq \sum_{l=1}^{d} \langle \nabla C(\mathbf{F})^l, \mathbf{V}^l - \mathbf{F}^l \rangle.$$

So,

$$\inf_{\mathbf{W} \in \mathcal{S}} \langle \nabla C(\mathbf{F}), \mathbf{W} - \mathbf{F} \rangle \geq d \langle \nabla C(\mathbf{F}), \mathbf{V}^{\hat{l}} - \mathbf{F}^{\hat{l}} \rangle. \tag{12}$$

Combining (11) and (12), we have

$$\langle \nabla C(\mathbf{F}), \mathbf{V}^{\hat{l}} - \mathbf{F}^{\hat{l}} \rangle \leq -\frac{\Delta C(\mathbf{F})}{d}. \tag{13}$$

Lemma 2 implies that

$$\Delta C(\mathbf{F}[\alpha, \mathbf{V}]) \leq \Delta C(\mathbf{F}) - \alpha \frac{\Delta C(\mathbf{F})}{d} + \frac{M}{2} \alpha^2. \tag{14}$$

By taking the minimum on the right hand side of (14) with respect to α on $[0, 1]$, we will get the desired result since $\Delta C(\mathbf{F})/dM \leq 1$ for $d \geq 1$. ■

Proof of Theorem 1. We will use the mathematical induction. First, consider the case of $m = 1$. Then,

$$\Delta C(\mathbf{F}_1) \leq M \leq d^2 M \leq \frac{2d^2 M}{m},$$

and so Theorem 1 is true for $m = 1$.

Next, suppose $\Delta C(\mathbf{F}_m) \leq 2d^2 M/m$ for $m \geq 2$. Lemma 3 implies

$$\Delta C(\mathbf{F}_{m+1}) \leq \Delta C(\mathbf{F}_m) - \frac{\Delta C(\mathbf{F}_m)^2}{2d^2 M}.$$

Since $x - x^2/(2d^2 M)$ is increasing on $[0, d^2 M]$ and $2d^2 M/m \leq d^2 M$ for $m \geq 2$, we have

$$\begin{aligned} \Delta C(\mathbf{F}_{m+1}) &\leq \frac{2d^2 M}{m} - \frac{(2d^2 M)^2}{2d^2 M m^2} \\ &= \frac{2d^2 M}{m} - \frac{2d^2 M}{m^2} \leq \frac{2d^2 M}{m+1}, \end{aligned}$$

which completes the proof. ■

Remark. In the standard LASSO framework, that is $d = 1$ and the cost function $C(\mathbf{F})$ is a function of $\sum_{k=1}^{p} \mathbf{f}_k$, we explained in Section 3 that the proposed gradient descent algorithm is the same as the feasible directional method, which finds the optimal convex combinations of vectors $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_p\} \cup \{-\mathbf{z}_1, \ldots, -\mathbf{z}_p\}$. The convergence rates of algorithms of finding the optimal convex combination of elements in the set $\mathcal{Z}$ have been studied by many authors including Jones (1992), Barron (1993) and Zhang (2003). In particular, Zhang (2003) considered the sequential greedy approximation algorithm which updates the current convex combination $\mathbf{h}_m$ to the new one $\mathbf{h}_{m+1}$ by $(1-\hat{\alpha})\mathbf{h}_m + \hat{\alpha}\hat{\mathbf{v}}$ where

$$(\hat{\alpha}, \hat{\mathbf{v}}) = \arg \min_{\mathbf{v} \in \mathcal{Z}, \alpha \in [0,1]} C\left((1-\alpha)\mathbf{h}_m + \alpha \mathbf{v}\right).$$

He proved that $\Delta C(\mathbf{h}_m) = O(m^{-1})$. Since the sequential greedy optimization algorithm is the fastest sequential algorithm, Theorem 1 implies that the convergence rate of the gradient descent algorithm for the generalized LASSO is almost optimal, at least for the standard LASSO model. Note that in the sequential greedy approximation algorithm, finding $(\hat{\alpha}, \hat{\mathbf{v}})$ is not easy because $C\left((1-\alpha)\mathbf{h}_m + \alpha \mathbf{v}\right)$ is not jointly convex in α and $\mathbf{v}$.

Remark. The bound in Theorem 1 implies that the convergence rate of the proposed algorithm does not depend on the dimension of inputs. Hence, we expect that the proposed algorithm works well with large dimensional data. This important feature of the proposed algorithm is also confirmed empirically in Section 5.

5. Simulation

In this section, we compare the performance of the gradient descent algorithm with the conventional QP algorithm by simulation. We first consider the multiple linear regression model given as

$$y = \beta_0 + \beta_1 z_1 + \cdots + \beta_p z_p + \epsilon.$$

For experiments, we choose the number of input variables of $p = 10, 20, 30, 50$ and 100. The inputs z_i are generated independently from the standard normal distribution. Also, ϵ follows the standard normal distribution. We set $\beta_0 = 1.0, \beta_1 = -0.5, \beta_2 = 1.0, \beta_4 = 0.5$ and other remaining coefficients are set to zero. We repeat the experiment 100 times with the fixed sample size 50. When β_0 is to be estimated, as we explained in Section 2, we put the constraint on β_0 as $|\beta_0| \le \lambda_0$ and let $\mathbf{x}_i^0 = 1$. Then, we apply the gradient descent algorithm with the augmented inputs $\mathbf{x}_i^* = (\mathbf{x}_i^0, \mathbf{x}_i)$. If λ_0 is sufficiently large that the LASSO estimate $\hat{\beta}_0$ satisfies $|\hat{\beta}_0| \le \lambda_0$, we get the desired estimate. For the choice of λ_0, we recommend $\lambda_0 = \eta \hat{\beta}_0^*$ where $\eta > 1$ and $\hat{\beta}_0^*$ is the minimizer of $\sum_{i=1}^{n} \mathcal{L}(y_i, \beta_0)$. Our experience confirms that $\eta \in [2, 3]$ works well.

Table 1 shows the number of iterations for varying number of inputs (p) and different constraints ($\lambda = 0.4,\ 0.8.\ 1.2,\ 1.6$ and 2.0) from the gradient descent algorithm. This confirms the theoretical result in Theorem 1 that the convergence speed of the gradient decent algorithm does not depend on the number of input variables.

In order to compare the gradient descent algorithm with the QP algorithm, we use `l1ce()` function in R system which implemented by Lokhorst et al. (1999) applying the algorithm of Osborne et al. (2000). The resulting deviances (the empirical risk divided by the sample size) of the two LASSO algorithms are summarized in Table 2. The deviance of the gradient descent algorithm is slightly larger than that of the QP. This is because the gradient algorithm is an approximation method in the sense that it converges to the optimum when the number of iteration goes to infinity. Actually, we stop the algorithm when the decrease of the deviance is small enough (we use 10^{-3}). The deviance of the gradient descent algorithm can be reduced more by iterating the algorithm more. Note, however, that the number of nonzero coefficients in Table 3 are almost identical.

For the multiple logistic regression model, we consider the following model :

$$f(\mathbf{z}) = \log\left(\frac{P(y=1|\mathbf{z})}{1-P(y=1|\mathbf{z})}\right) = 1.0 - 2z_1 + z_2 + 0.5z_3.$$

All the other experimental settings are the same as those for the multiple regression. The average number of iterations of the gradient descent algorithm is in Table 4, and the deviances and the non-zero coefficients are found in Table 5 and 6. These results are as similar as those of the multiple linear regression model.

Our simulation results show that the gradient descent algorithm for the generalized LASSO finds the (almost) optimum with much less computation than the QP method. Moreover, its convergence rate does not depend on the number of inputs.

6. Analysis of Pima Indians Diabetes dataset using the structural sparse kernel machine

One important advantage of the the proposed algorithm is to be able to analyze the model with multiple constraints such as the structural sparse kernel machine in Example 3 of Section 3. In this section, we analyze the Pima Indians Diabetes data using the structural sparse kernel machine. The data set, which is available in the UCI machine learning repository (http://www.ics.uci.edu/∼mlearn /MLRepository.html), has 768 observations with 8 input variables and a binary-class output variable. The 8 input variables and one output variable are follows.

1. `preg` : Number of times pregnant
2. `glu` : Plasma glucose concentration (glucose tolerance test)
3. `pres` : Diastolic blood pressure (mmHg)
4. `tri` : Triceps skin fold thickness(mm)
5. `insu` : 2-Hour serum insulin (mu U/ml)
6. `mass` : Body mass index (weight in kg/(height in m)^2)
7. `pedi` : Diabetes pedigree function
8. `age` : Age (years)
9. `diabetes` : Class output variable (500 "−" and 268 "+" tests for diabetes)

p	$\lambda = 0.4$	$\lambda = 0.8$	$\lambda = 1.2$	$\lambda = 1.6$	$\lambda = 2.0$
10	4.03 ± 0.13	15.23 ± 3.02	93.21 ± 6.97	181.46 ± 4.34	233.84 ± 3.71
20	4.04 ± 0.13	15.42 ± 3.04	92.53 ± 6.97	180.16 ± 4.14	234.52 ± 3.90
30	4.15 ± 0.15	15.65 ± 2.98	91.06 ± 6.89	179.32 ± 4.16	234.32 ± 3.66
50	4.21 ± 0.15	15.49 ± 2.89	90.58 ± 6.85	182.40 ± 4.16	235.81 ± 3.41
100	4.26 ± 0.14	14.94 ± 2.66	90.16 ± 6.83	181.52 ± 4.17	245.34 ± 3.39

Table 1: Average number of iterations for the gradient descent algorithm on linear regression

p	$\lambda = 0.4$	$\lambda = 0.8$	$\lambda = 1.2$	$\lambda = 1.6$	$\lambda = 2.0$
			Gradient method		
10	133.19 ± 3.033	105.76 ± 2.457	85.03 ± 1.997	69.56 ± 1.642	58.15 ± 1.363
20	133.19 ± 3.033	105.76 ± 2.457	85.00 ± 1.997	69.42 ± 1.644	57.72 ± 1.372
30	133.18 ± 3.033	105.71 ± 2.458	84.91 ± 1.998	69.23 ± 1.649	57.30 ± 1.382
50	133.15 ± 3.037	105.65 ± 2.465	84.74 ± 2.011	68.83 ± 1.668	56.56 ± 1.403
100	133.14 ± 3.040	105.56 ± 2.468	84.45 ± 2.018	68.20 ± 1.683	55.37 ± 1.423
			QP method		
10	133.19 ± 3.033	105.74 ± 2.459	84.85 ± 2.003	69.17 ± 1.644	57.64 ± 1.363
20	133.19 ± 3.033	105.73 ± 2.459	84.81 ± 2.003	69.01 ± 1.647	57.19 ± 1.373
30	133.18 ± 3.033	105.69 ± 2.460	84.72 ± 2.004	68.82 ± 1.651	56.77 ± 1.384
50	133.15 ± 3.037	105.63 ± 2.468	84.55 ± 2.017	68.42 ± 1.670	56.03 ± 1.405
100	133.14 ± 3.040	105.54 ± 2.471	84.25 ± 2.024	67.78 ± 1.685	54.85 ± 1.424

Table 2: Average deviances of the gradient descent algorithm and QP algorithm on linear regression

p	$\lambda = 0.4$	$\lambda = 0.8$	$\lambda = 1.2$	$\lambda = 1.6$	$\lambda = 2.0$
			Gradient method		
10	2.87 ± 0.06	3.54 ± 0.07	4.19 ± 0.08	4.86 ± 0.09	5.57 ± 0.09
20	2.88 ± 0.07	3.58 ± 0.07	4.38 ± 0.10	5.35 ± 0.13	6.63 ± 0.15
30	2.91 ± 0.07	3.65 ± 0.09	4.56 ± 0.13	5.79 ± 0.16	7.47 ± 0.18
50	2.95 ± 0.08	3.79 ± 0.10	4.81 ± 0.15	6.35 ± 0.21	8.46 ± 0.24
100	3.01 ± 0.08	4.10 ± 0.15	5.42 ± 0.22	7.51 ± 0.27	10.26 ± 0.33
			QP method		
10	2.87 ± 0.06	3.54 ± 0.07	4.19 ± 0.08	4.95 ± 0.08	5.66 ± 0.09
20	2.88 ± 0.07	3.58 ± 0.07	4.39 ± 0.10	5.43 ± 0.13	6.80 ± 0.16
30	2.91 ± 0.07	3.64 ± 0.08	4.61 ± 0.14	5.93 ± 0.17	7.61 ± 0.19
50	2.95 ± 0.08	3.78 ± 0.10	4.85 ± 0.15	6.49 ± 0.21	8.68 ± 0.24
100	3.01 ± 0.08	4.12 ± 0.16	5.45 ± 0.22	7.64 ± 0.27	10.48 ± 0.33

Table 3: The number of non-zero coefficients of the gradient descent algorithm and QP algorithm on linear regression

p	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 4$	$\lambda = 5$
10	9.79 ± 1.16	63.19 ± 2.57	99.87 ± 2.25	125.87 ± 2.53	140.93 ± 2.63
20	10.31 ± 1.15	65.41 ± 2.51	105.40 ± 2.10	135.92 ± 2.04	164.55 ± 2.38
30	10.38 ± 1.07	65.68 ± 2.45	109.57 ± 2.19	142.53 ± 2.30	173.34 ± 2.74
50	11.27 ± 1.08	66.99 ± 2.43	112.61 ± 2.36	150.18 ± 2.25	178.56 ± 2.59
100	12.59 ± 1.23	70.57 ± 2.59	119.53 ± 2.25	155.03 ± 2.00	185.11 ± 2.53

Table 4: Average number of iterations for the gradient descent algorithm on logistic regression

p	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 4$	$\lambda = 5$
			Gradient method		
10	25.78 ± 0.13	21.52 ± 0.19	18.93 ± 0.22	17.22 ± 0.25	16.06 ± 0.28
20	25.71 ± 0.13	21.23 ± 0.18	18.30 ± 0.20	16.18 ± 0.22	14.58 ± 0.23
30	25.68 ± 0.13	21.05 ± 0.17	17.90 ± 0.19	15.57 ± 0.20	13.73 ± 0.21
50	25.62 ± 0.13	20.78 ± 0.16	17.37 ± 0.18	14.75 ± 0.18	12.66 ± 0.18
100	25.46 ± 0.12	20.29 ± 0.14	16.52 ± 0.15	13.58 ± 0.16	11.24 ± 0.15
			QP method		
10	25.77 ± 0.13	21.38 ± 0.18	18.70 ± 0.22	16.93 ± 0.25	15.71 ± 0.28
20	25.70 ± 0.13	21.08 ± 0.17	18.06 ± 0.20	15.87 ± 0.22	14.20 ± 0.23
30	25.67 ± 0.13	20.90 ± 0.17	17.66 ± 0.19	15.25 ± 0.20	13.35 ± 0.21
50	25.61 ± 0.12	20.63 ± 0.16	17.12 ± 0.18	14.44 ± 0.18	12.28 ± 0.18
100	25.44 ± 0.11	20.14 ± 0.14	16.27 ± 0.15	13.27 ± 0.16	10.86 ± 0.15

Table 5: Average deviances of the gradient descent algorithm and QP algorithm on logistic regression

p	$\lambda = 1$	$\lambda = 2$	$\lambda = 3$	$\lambda = 4$	$\lambda = 5$
			Gradient method		
10	3.02 ± 0.09	4.78 ± 0.12	6.35 ± 0.14	7.34 ± 0.15	8.26 ± 0.14
20	3.43 ± 0.12	5.96 ± 0.17	8.29 ± 0.19	10.03 ± 0.20	11.43 ± 0.23
30	3.70 ± 0.13	6.77 ± 0.20	9.47 ± 0.22	11.93 ± 0.22	13.71 ± 0.24
50	4.20 ± 0.16	7.92 ± 0.22	11.09 ± 0.25	14.06 ± 0.27	16.17 ± 0.29
100	4.87 ± 0.18	9.69 ± 0.24	13.67 ± 0.28	16.71 ± 0.30	19.31 ± 0.33
			QP method		
10	3.04 ± 0.09	4.94 ± 0.13	6.52 ± 0.15	7.69 ± 0.15	8.61 ± 0.14
20	3.50 ± 0.13	6.18 ± 0.18	8.71 ± 0.19	10.42 ± 0.21	11.90 ± 0.23
30	3.79 ± 0.14	6.96 ± 0.21	10.05 ± 0.23	12.29 ± 0.22	14.10 ± 0.23
50	4.31 ± 0.16	8.11 ± 0.22	11.54 ± 0.25	14.65 ± 0.27	16.66 ± 0.29
100	5.27 ± 0.20	10.12 ± 0.24	14.12 ± 0.29	17.17 ± 0.31	19.61 ± 0.31

Table 6: The number of non-zero coefficients of the gradient descent algorithm and QP algorithm on logistic regression

The model starts with the main effects and second order interaction terms as in (5). We set two tuning parameters (λ_1 and λ_2) to be equal, that is, $\lambda_1 = \lambda_2 = \lambda$ and choose $\lambda = 6$, which is selected by the 5-fold cross-validation. The Gaussian kernel with the scale parameter reciprocally proportional to the dimension of inputs is used.

Figure 2 depicts the change of the deviance on the number of the iteration, which shows that the proposed gradient descent algorithm converges fairly fast in the early iterations. Even after 50 iterations, the deviance is very close to the optimum. Though not presented, we report that this feature is observed in all the data sets we analyzed. So, we can save computing time more by stopping earlier with sacrificing little accuracy.

Figure 3 presents the L_1 norms of the selected components which are considered as the importance measure of the input variables. Here, L_1 norm of $f_j(z_j)$ is defined by $\sum_{i=1}^n |\sum_{l=1}^n c_{lj} K_j(z_{lj}, z_{ij})| /n$, and L_1 norm of the second order interaction term is defined similarly. One interesting finding is that the interaction term `tri*ins` is selected as the third most important feature. It is well known that finding important (nonlinear) interaction terms is hard. The combination of the kernel idea and proposed algorithm makes it simpler.

7. Gene Selection using the sparse muticlass logistic regression

In this section, we apply the proposed algorithm for selecting relevant genes by analyzing microarray data with the sparse multiclass logistic regression model. Let $\mathbf{z}_i = (z_{i1}, \ldots, z_{ip})$ be gene expression levels from the i-th examples, $i = 1, \ldots, n$, and for each example, there is a corresponding label $y_i \in \{1, 2, \ldots, J\}$ representing the specific type of cancer. The model we consider is

$$\Pr(y_i = j|\mathbf{z}_i) = \frac{\exp\left(f_j(\mathbf{z}_i)\right)}{1 + \sum_{m=1}^{J-1} \exp\left(f_m(\mathbf{z}_i)\right)}$$

for $j = 1, \ldots, J-1$, and

$$\Pr(y_i = J|\mathbf{z}_i) = \frac{1}{1 + \sum_{m=1}^{J-1} \exp\left(f_m(\mathbf{z}_i)\right)}$$

where

$$f_j(\mathbf{z}_i) = \theta_0^{(j)} + \rho_1 \theta_1^{(j)} z_{i1} + \cdots + \rho_p \theta_p^{(j)} z_{ip}.$$

For identification of the model, we let $\sum_{j=1}^{J-1} |\theta_k^{(j)}| = 1$ and $\rho_k \geq 0$ for $k = 1, \ldots, p$. We estimate $\theta_k^{(j)}$s as well as ρ_ks by maximizing the log-likelihood given in (6) with the constraint $\sum_{k=1}^p \rho_k \leq \lambda$. Since the solution of ρ_ks are sparse, we can select relevant genes whose ρs are not zero.

The above model can be embedded into the generalized LASSO set-up by letting $\beta_k^{(j)} = \rho_k \theta_k^{(j)}$ with the constraint

$$\sum_{j=1}^{J-1} \sum_{k=1}^{p} |\beta_k^{(j)}| \leq \lambda. \tag{15}$$

Then, the estimates of ρs as well as θs can be recovered from the estimates of βs by $\rho_k = \sum_{j=1}^{J-1} |\beta_k^{(j)}|$ and $\theta_k^{(j)} = \beta_k^{(j)} / \rho_k$.

Now, we analyze the small round blue cell tumors of children data set in Khan et al. (2001), which consists of the expression levels of 2,308 genes and 4 class labels - 4 cancer

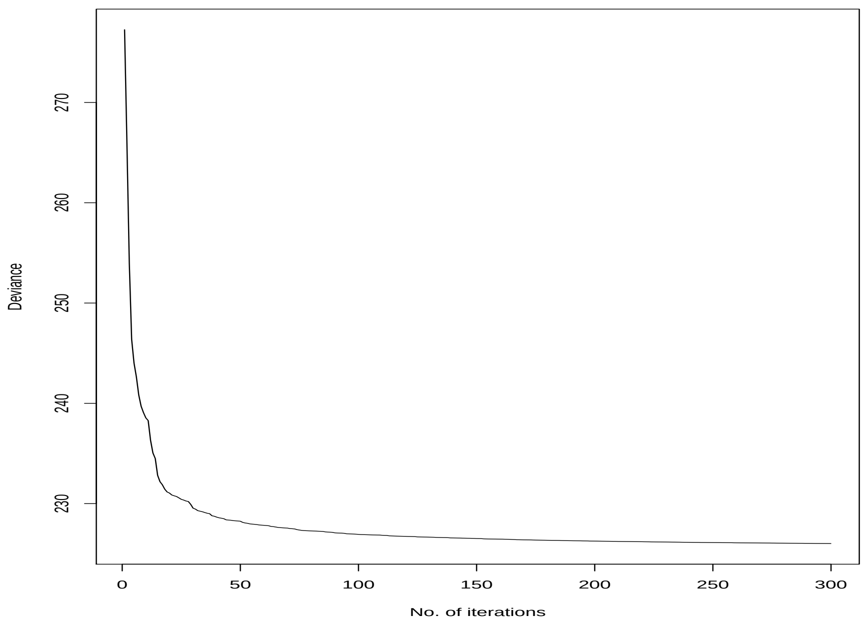


Figure 2: Deviance vs. the number of iterations for the gradient descent algorithm on the structural sparse kernel machine

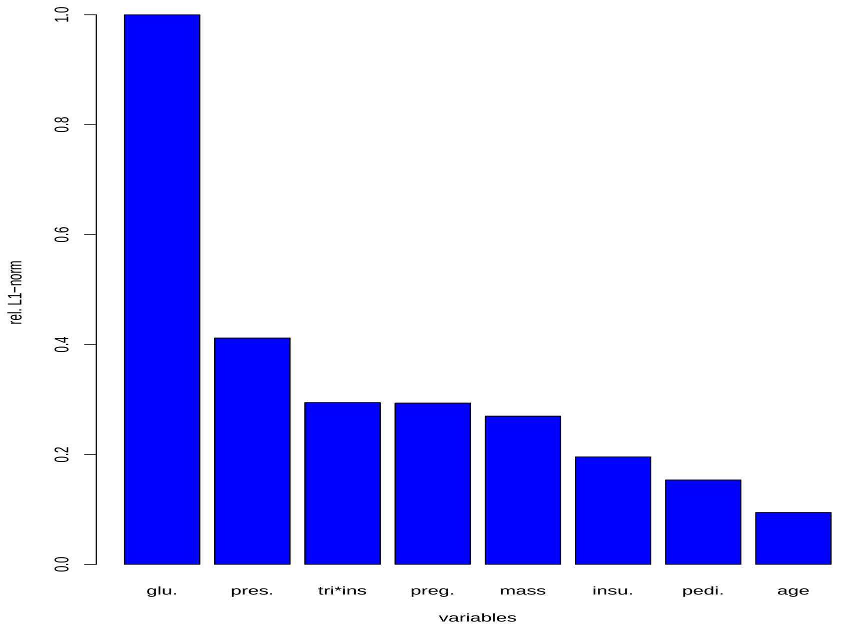


Figure 3: The L_1-norms of the selected components using the gradient descent algorithm on the structural sparse kernel machine

types including neuroblastoma (NB), rhabdomyosarcoma (RMS), non- Hodgkin lymphoma (NHL) and the Ewing family of tumors (EWS). The data set is divided into training set with 63 examples and test set with 20 examples. All the expression levels are standardized to be centered at 0 and have unit variance.

The regularization parameter λ is chosen by the leave-one-out cross validation. We used the two measurement for selecting the optimal λ - CV_1 and CV_2 defined as

$$CV_1 = \sum_i \left[1 - I\left(y_i = \arg\max_j f_j^{[-i]}(\mathbf{z}_i) \right) \right], \tag{16}$$

$$CV_2 = \sum_i \left[-\sum_j I(y_i = j) f_j^{[-i]}(\mathbf{z}_i) + \log\left(\exp(\sum_j f_j^{[-i]}(\mathbf{z}_i)) \right) \right], \tag{17}$$

where $f_j^{[-i]}$ is the estimate of f_j based on the cross validation data set without the i-th example. That is, CV_1 uses the 0-1 loss and CV_2 uses the negative log-likelihood loss. Though CV_1 is a standard measurement for the cross validation, we tried CV_2 because it behaves more nicely as a function of λ. Figure 4 shows the results over $\lambda = 0.1$, 0.2, 0.5 ,1, 2, 5, 10, 20, 50, 100, 200, and 500. While CV_1 has multiple minima around λ in between 20 and 50, CV_2 has the unique minimum at $\lambda = 50$. We conjecture that this is because the variation of CV_1 is larger than that of CV_2. Also, the test errors of the models selected by CV_1 and CV_2 are the same - one out of 20 test set examples. Hence, we recommend CV_2 when the main purpose is to select the value of the regularization parameter but not to estimate the generalization error.

Reliability of the selected genes by the generalized LASSO is another important issue. With $\lambda = 20$ selected by CV_1, 72 genes were selected, and with $\lambda = 50$ selected by either CV_1 or CV_2, 102 genes were selected. Considering the fact that all inputs were standardized, we can regard the coefficient ρ_k as the importance measure of the k-th gene. We compare ρ_k with the gene rank based on the marginal F-ratio, which is a popular filtering technique (Dudoit and Speed, 2002; Lee et al., 2004). Figure 5 shows the plots of ρs vs. the gene ranks by the marginal F-ratios with $\lambda = 20$ and $\lambda = 50$, respectively. In general, the sparse multilcass logistic model yields large values of ρ_k to genes whose ranks based on the marginal F-ratio are higher. However, surprisingly, one gene whose marginal rank is lower than 500 is selected with relatively large ρ_k by the the sparse multilcass logistic model. The standard procedure for constructing a classification model with gene expression data is first to select small to medium number of genes using a filtering technique such as the marginal F-ratio, and then to construct a predictive model based on the pre-selected genes. The main reason for using this two-stage procedure is that when the number of candidate genes are too large, the most of classification algorithms have computational problems. However, our results find negative evidence for this standard two-stage procedure, and implies that efficient computing algorithms are essential for constructing a good predictive model with gene expression data set. The proposed algorithm is, of course, one of such algorithms.

8. Conclusions

In this paper, we proposed the gradient descent method for the generalized LASSO. The proposed algorithm is computationally simpler and faster than the conventional QP method.

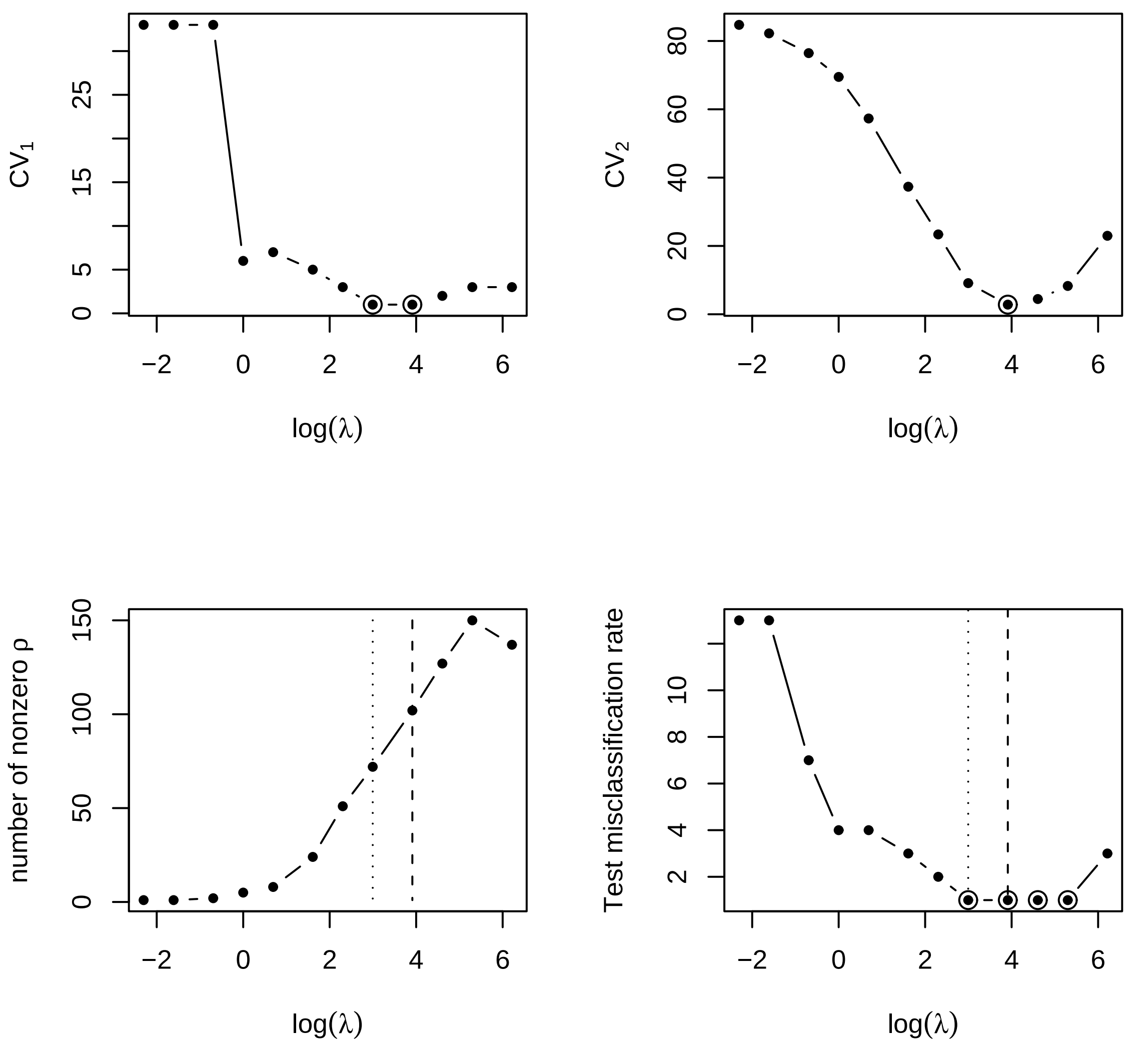
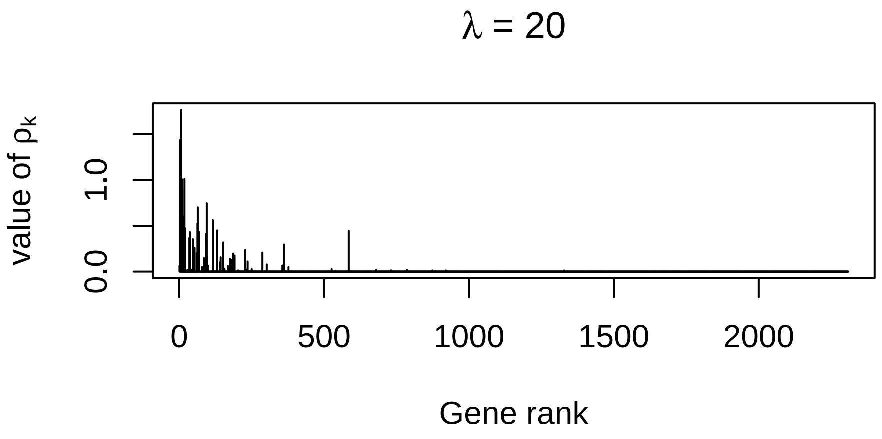


Figure 4: Results of the small round blue cell tumors of children data set by the sparse multiclass logistic model. Left Top: The CV_1 plot with its minimizers marked by circles. Right Top: The CV_2 plot with its minimizer marked by circle. Left Bottom: The number of nonzero ρ_k. The dotted line is drawn at the $\lambda = 20$ chosen by CV_1. The dashed line is drawn at the $\lambda = 50$ chosen by either CV_1 or CV_2. Right Bottom: The number of misclassified examples with there minimizers marked by circles.



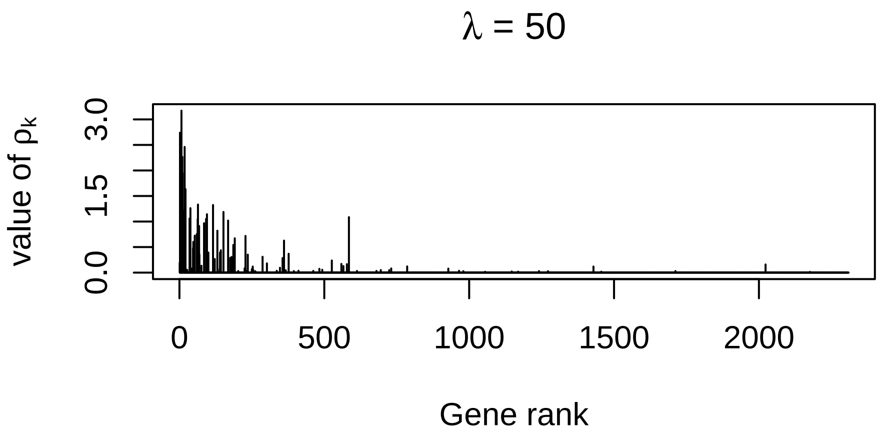


Figure 5: The estimated value of ρ_k sorted by the rank of the marginal F-ratio. Top: the result with $\lambda = 20$ chosen by CV_1. Bottom: the result with $\lambda = 50$ chosen by either CV_1 or CV_2.

We also showed theoretically as well as empirically that the proposed algorithm converges fairly fast and gives reliable results

One interesting feature of the proposed algorithm is that the convergence rate is independent on the dimension of inputs. The convergence rate is important since less iteration gives more sparse solutions. Hence, the proposed algorithm is well suited with problems with large dimensional inputs such as the sparse kernel machine and the analysis of the gene expression data set. In the analysis of the Pima Indians Diabetes data set, the algorithm converges to the near optimum only after 50 iterations. This is surprising since the number of components is 36. Also, the final model, which contains only 8 components out of 36, which is very sparse. Another advantage of our algorithm is that it does not require any additional matrix operations in the preprocessing step which is necessary when the design matrix is singular.

In the gene expression data, we can select around 100 genes out of more than 2,000 genes using the proposed algorithm with the sparse multiclass logistic regression model. In particular, our approach does not need any pre-filtering methods. Also, our result shows a possibility that a pre-filtering method can delete a significant gene. Within our knowledge, our analysis is the first of its kind for gene expression data analysis.

One problem of the proposed algorithm is that the convergence speed is rather slow at the near optimum. Figure 2 represents the typical behavior of the deviance. That is, the deviance is dropped very fast in the early stage, and then it decreases slowly. This is one reason why the deviances of the proposed algorithm are slightly larger than those of the QP in our simulation results. Accelerating the convergence speed at the near optimum is worth pursuing.

Acknowledgments

This research was supported in part by US Air Force Research Grant F62562-03-P-0658 and in part by KOSEF through Statistical Research Center for Complex Systems at Seoul National University, Korea.

References

Sergei Bakin. *Adaptive regression and model selection in data mining problem.* PhD thesis, Australian National University, Australia, 1999.

Andrew R. Barron. A universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. Inform. Theory*, 39:930–945, 1993.

Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, Massachusetts, 1999.

Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

Fridlyand J. Dudoit, S. and T. Speed. Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association*, 97:77–87, 2002.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

Yves Grandvalet and Stéphane Canu. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. In Micael Kearns, Sara Solla, and David Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 445–451. MIT press, 1999.

Steve R. Gunn and Jaz S. Kandola. Structural modelling with sparse kernels. *Machine Learning*, 48:115–136, 2002.

L. K. Jones. A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *Annals of Statistics*, 20:608–613, 1992.

J. Khan, J. Wei, M. Ringner, L.Saal, M. Ladanyi, F. Westermann, F. Berthold, M. Schwab, C. Atonescu, C. Peterson, and P. Meltzer. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nature Medicine*, 7:673–679, 2001.

George S. Kimeldorf and Grace Wahba. Some results on Tchebychffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1972.

B. Krishnapuram, L. Carlin, M.A.T. Figueiredo, and A. Hartemink. Learning sparse classifier: Multi-class formulation, fast algorithms and generalization bounds. Technical report, ISDS, Duke university, 2004.

Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99:67–81, 2004.

J. Lokhorst, B. A. Turlach, and W. N. Venables. Lasso2*: An S-plus library to solve regression problems while imposing an L1 constraint on the parameters, 1999.

M. R. Osborne, Brett Presnell, and B. A. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20(3):389–404, 2000.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremetal feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3: 1333–1356, 2003.

V. Roth. The generalized lasso. *IEEE transactions on neural networks*, 15:16–28, 2004.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *J.R. Statist. Soc. (B)*, 58:267–288, 1996.

Hao Helen Zhang, Grace Wahba, Yi Lin, Meta Voelker, Michael Ferris, Ronald Klein, and Barbara Klein. Variable selection and model building via likelihood basis pursuit. Technical Report 1059r, Department of Statistics, University of Wisconsin, Madison, WI, 2003.

Tong Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Trans. Inform. Theory*, 49:682–691, 2003.